What Is Claimed Is:

1. A method of controlling a processor, comprising:

allocating a configuration field within an instruction to select a configuration register; and

executing the instruction and a configuration in the configuration register selected by the configuration field.

2. The method of claim 1, before executing the configuration and the instruction, further comprising loading the configuration register with the configuration.

3. The method of claim 2, wherein loading the configuration is performed with a separate instruction.

4. The method of claim 3, wherein the separate instruction includes a displacement field indicating a location of the configuration within a memory relative to a location of the instruction within the memory.

5. The method of claim 1, before executing the instruction and the configuration, further comprising:

decoding the instruction into instruction controls; and

decoding the configuration into configuration controls.

6. The method of claim 5, further comprising:

providing the instruction controls and the configuration controls to one or more execution units;

processing data according to the instruction and configuration controls; and

generating a result according to the instruction and configuration controls.

7. The method of claim 6, further comprising writing the result to a register.

8. The method of claim 6, further comprising writing the result to a memory.

9. The method of claim 6, further comprising providing the result to an execution unit.

10. The method of claim 1, wherein the instruction further comprises a modification field, the method further comprising decoding the modification field to alter the configuration controls decoded from the configuration.

11. The method of claim 1, wherein executing the instruction and the configuration further comprises performing an operation on scalar data.

12. The method of claim 1, wherein executing the instruction and the configuration further comprises performing an operation on vector data.

13. The method of claim 1, wherein executing the instruction and the configuration further comprises performing operations on both scalar and vector data.

14. A method of controlling a reconfigurable processor, comprising:
executing a first instruction that loads a configuration into a configuration register;
executing a second instruction that references the configuration register; and
executing the configuration in the configuration register referenced by the second instruction.

15. The method of claim 14, wherein executing the first instruction loads a plurality of configurations into respective configuration registers, wherein one of the plurality of configurations is loaded into a configuration register.

16. The method of claim 15 wherein the configuration and the first instruction are stored in a memory, and wherein the first instruction includes a displacement field indicating a location in the memory of the configuration relative to the first instruction.

17. The method of claim 14, wherein an application program issues the first instruction.

18. The method of claim 14, wherein a compiler generates the first instruction.

19. The method of claim 14, wherein executing the second instruction and the configuration further comprises retrieving operands requested by the second instruction and the configuration.

20. The method of claim 19, wherein the second instruction provides the operands to the configuration.

21. The method of claim 19, wherein a register provides the operands to the configuration.

22. The method of claim 19, wherein the second instruction includes an immediate value field, the second instruction being executed with values stored in the immediate value field.

23. The method of claim 19, wherein the second instruction includes an immediate value field, the configuration being executed with values stored in the immediate value field.

24. The method of claim 14, further comprising:

decoding controls from the second instruction and the configuration; and

processing data according to the decoded controls with one or more execution units in parallel.

25. The method of claim 24, further comprising generating one or more results with the one or more execution units.

26. The method of claim 25, further comprising writing the one or more results to a register.

27. The method of claim 25, further comprising storing the one or more results to a memory.

28. The method of claim 25, further comprising providing the one or more results to respective execution units.

29. The method of claim 14, further comprising pre-loading a second configuration register with a configuration while the configuration previously loaded in the first configuration register executes.

30. The method of claim 14, further comprising stalling the second instruction while the referenced configuration register is being loaded with a configuration.

31. The method of claim 14, wherein the first instruction, the second instruction, and the configuration are executed as part of an application program.

32. The method of claim 14, wherein executing the second instruction and the configuration includes performing an operation on scalar data.

33. The method of claim 14, wherein executing the second instruction and the configuration includes performing an operation on vector data.

34. The method of claim 14, wherein executing the second instruction and the configuration includes performing an operation on scalar data and performing an operation on vector data.

35. A processing system, comprising:

means for referencing, in an instruction, a configuration register storing a configuration; and

means for executing the instruction and the configuration in the configuration register referenced by the instruction.

36. A processing system, comprising:

means for executing a first instruction that loads a configuration into a configuration register; and

means for executing a second instruction and the configuration, the second instruction referencing the configuration register containing the configuration.

37. A processing system, comprising:

a plurality of configuration registers, each configuration register designed to store a configuration,

wherein an instruction selects one of the plurality of configuration registers based on a value in a configuration field of the instruction, and

wherein the instruction and the configuration stored in the selected configuration register are both executed.

38. The system of claim 37, further comprising:

an instruction decode unit configured to determine the instruction controls; and

a configuration decode unit configured to determine controls of the configuration stored in the configuration register selected by the instruction.

39. The system of claim 38, wherein an execution unit is controlled by the configuration controls in the absence of the instruction controls.

40. The system of claim 38, the instruction further comprising a modification field configured to modify the configuration controls.

41. The system of claim 40, further comprising a component to implement the modified configuration controls.

42. The system of claim 41, wherein the component implementing the modified configuration controls comprises a multiplexer.

43. The system of claim 41, wherein the component implementing the modified configuration controls comprises a logic circuit.

44. The system of claim 41, wherein the component implementing the modified configuration controls comprises a function.

45. The system of claim 40, wherein different configuration controls are decoded from a single configuration as a result of the modified configuration controls.

46. The system of claim 40, wherein the modified configuration controls result in an altered operation code of an execution unit.

47. The system of claim 40, wherein the modified configuration controls result in an altered register number.

48. The system of claim 40, wherein the modified configuration controls result in an inhibited register write function.

49. The system of claim 40, wherein the modified configuration controls result in a cleared register.

50. The system of claim 40, wherein the modified configuration controls result in an incremented counter.

51. The system of claim 38, further comprising a register configured to provide operands requested by the instruction and configuration controls.

52 The system of claim 51, wherein the register comprises a register file.

53. The system of claim 51, further comprising an execution unit configured to process the operands from the register according to the instruction and configuration controls.

54. The system of claim 53, wherein the execution unit comprises an arithmetic logic unit

55. The system of claim 53, wherein the execution unit comprises a shift unit.

56. The system of claim 53, wherein the execution unit comprises a multiply unit.

57. The system of claim 53, wherein the execution unit is configured to generate a result.

58. The system of claim 57, wherein the result is stored to an internal register.

59. The system of claim 57, wherein the result is stored to a register file.

60. The system of claim 57, wherein the result is stored to a predicate register.

61. The system of claim 57, wherein the result is stored to a memory.

62. The system of claim 57, wherein the result is provided to an execution unit.

63. The system of claim 37, wherein a plurality of configurations are loaded from a memory to respective configuration registers with a load instruction, the load instruction comprising:

one or more operation code fields that control loading of the configurations;

a configuration field identifying a configuration register that is loaded from the memory; and

a displacement indicating a memory address storing a configuration to be loaded into the configuration register.

64. The system of claim 63, wherein a path to fetch the instruction that selects one of the plurality of configuration registers is also used to load the configuration into the configuration register.

65. The system of claim 63, wherein the configuration registers are loaded while previously loaded configurations are executed.

66. The system of claim 37, wherein the instruction and configuration controls execute an operation on scalar data.

67. The system of claim 37, wherein the instruction and configuration controls execute an operation on vector data.

68. The system of claim 37, wherein the instruction and configuration controls execute an operation on vector and scalar data.

69. A vector processing system, comprising:

a plurality of configuration registers, each configuration register configured to store a configuration,

wherein an instruction selects one of the plurality of configuration registers based on a value in a configuration field of the instruction, and

wherein the instruction and the configuration stored in the selected configuration register are both executed;

a vector register file configured with one or more write and read ports to store one or more vectors, each vector including a plurality of vector elements; and

a vector address unit configured to provide an address of a vector element to be processed by the instruction and the configuration to the vector register file.

70. The system of claim 69, wherein a vector element in an address identified by the vector address unit is provided through a read port of the vector register file and processed according to the instruction and configuration controls.

71. The system of claim 69, wherein a vector element is provided through a write port and stored to an address of the vector register file identified by the vector address unit according to the instruction and configuration controls.

72. The system of claim 69, each of the vector address units further comprising:
a register configured to store a current address of an element of the vector; and
an adder configured to add a stride to the current address,
wherein, for each vector element, the current address is incremented by the stride to identify an address of the next vector element to be processed.

73. The system of claim 72, wherein the stride comprises an implicit stride.

74. The system of claim 72, further comprising a stride register, wherein the stride comprises an address stride provided by the stride register.

75. The system of claim 72, further comprising a multiplexer configured to select an address of a vector element provided to the vector register file.

76. The system of claim 75, wherein the vector element address is the current address.

77. The system of claim 75, wherein the vector element address is provided by an immediate value of the instruction.

78. The system of claim 75, wherein the vector element address is an address from another register.

79. The system of claim 72, the vector address unit further comprising a register configured to store a start address of the vector.

80. The system of claim 72, the vector address unit further comprising a register configured to store a frame stride that increments the start address with the adder after all of the elements of a first vector have been processed, the incremented start address identifying the start address of a second vector.

81. The system of claim 72, further comprising:
a vector length register; and
a vector count register, wherein
the vector length register is configured to store a value representing a number of elements in the vector,

an initial value of the vector count register is the value of the vector length register, and

the vector count register is decremented beginning from the value in the vector length register for each vector element processed.

82. The system of claim 72, wherein a finite impulse response filter is implemented by processing vector data based on controls from the instruction and the configuration.

83. The system of claim 72, wherein a finite impulse response filter is implemented by processing vector and scalar data based on controls from the instruction and the configuration.

84. A method of implementing a vector processing system, comprising:

executing a first instruction that loads a configuration into a configuration register;

executing a second instruction and a configuration stored in a configuration register referenced by the second instruction;

processing elements of a first vector according to the second instruction and the configuration, wherein

a vector register stores elements of the first vector, and

a vector address unit provides an address to the vector register which stores the first vector elements selected by the second instruction and the configuration.

85. The method of claim 84, wherein processing elements of the first vector further comprises writing data to the identified address through a write port of the vector register file.

86. The method of claim 84, wherein processing elements of the first vector further comprises reading data from the identified address through a read port of the vector register file.

87. The method of claim 84, wherein processing elements of the first vector further comprises:

initializing a current address of the first vector with a start address;

processing a first element of the first vector referenced by the current address with the instruction and configuration.

88. The method of claim 87, further comprising:

incrementing the current address with an address stride, wherein the incremented current address represents an address of a second element of the first vector; and

processing the second element referenced by the incremented current address.

89. The method of claim 88, for each successive element of the first vector, further comprising:

incrementing the previous current address with the address stride resulting in a new current address, wherein each successive new current address represents an address of a successive vector element; and

processing each successive vector element until all of the elements of the first vector have been processed.

90. The method of claim 89, further comprising identifying a start address of a second vector.

91. The method of claim 90, wherein identifying the start address of the second vector further comprises incrementing the start address of the first vector with a frame stride resulting in a second start address, wherein an initial value of a current address comprises the second start address.

92. The method of claim 91, further comprising processing the vector element referenced by the current address of the second vector.

93. The method of claim 92, for each successive vector element of the second vector, further comprising:

incrementing the previous current address with the address stride resulting in a new current address, wherein each successive new current address represents an address of a successive vector element of the second vector; and

processing each successive vector element until all of the elements of the second vector have been processed.

94. The method of claim 93, for each vector to be processed, further comprising:

identifying a start address of the vector;

processing a first element of the vector;

processing remaining successive elements of the vector by

incrementing the current address with an address stride resulting in successive current addresses;

processing corresponding successive elements referenced by the successive current addresses; and

after all of the elements of the vector have been processed, incrementing the start address by the frame stride to identify a start address of the next vector to be processed.

95. The method of claim 84, wherein a vector of data elements is loaded into the vector file prior to execution of the second instruction and the configuration.

96. The method of claim 84, wherein a vector of data elements is loaded into the vector file in parallel with execution of the second instruction and the configuration.

97. The method of claim 96, wherein the first instruction operates to process the first vector element by element by interlocking between the load port of the vector register file and the vector computation to process each element when it arrives in the vector register file.